Lab Exercise: Syntax

About Intertech

Thank you for choosing Intertech for your training from Udemy. The next page of this document will get you started with your lab if you'd like to skip ahead. Below is a brief overview of Intertech as well as a promo code for you for future live Intertech trainings.

Intertech (www.intertech.com) is the largest combined software developer **training** and **consulting** firm in Minnesota. Our unique blend of training, consulting, and mentoring has empowered technology teams in small businesses, Fortune 500 corporations and government agencies since 1991.

Our training organization offers live in-classroom and online deliveries, private on-site deliveries, and on-demand options such as the course in which you've enrolled from Udemy. We cover a broad spectrum of .NET, Java, Agile/Scrum, Web Development, and Mobile technologies. See more information on our training and search for courses by **clicking here**.

As a Udemy customer, you can save on any live in-classroom, live online, or onsite training with Intertech as well. Just use promo code "Udemy_Labs" when enrolling **and** save 35% on the course price.

We appreciate you choosing Intertech on Udemy for this course!

Lab Exercise

Syntax

Java's syntax is similar to that of many programming languages. In this lab, you will explore a few basic Java code blocks such as conditional statements, a loop, and even a switch statement. You will also explore some numeric operators.

Specifically, in this lab you will:

- Use conditionals to return an appropriate value from a method
- Use a switch statement to guide calculations appropriately
- Use one of Java's loops to iterate over a series of numbers
- Explore the use of arithmetic operators with Java primitives

Scenario

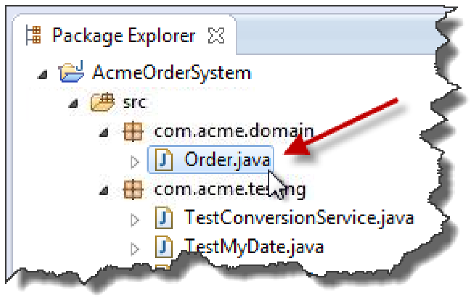
The Acme Order System development is rolling along nicely. The user community has put in some requests for utilities in the application. In this lab, you'll use your newfound knowledge of Java syntax to add a couple of utility methods in some of the already created classes.

Step 1: Add computing methods to order

At Acme, orders are often prioritized by job size. An order's job size is based on the quantity ordered and classified as small, medium, large, or extra-large. Below is a table outlining each of the job size classifications.

Order Quantity	Job Size Classification
<= 25	'S'
26–75	'M'
76–150	'L'
> 150	'X'

1.1 Add a method to compute an order's job size. In the Package Explorer view, double-click on the Order.java file in the com.acme.domain package to open the file in a Java editor.



1.2 Add a jobSize method. The method definition should look like the code below.

```
public char jobSize() {
  // ... add your code here to compute and
  // return the job size for an order.
}
```

1.3 Add a method to calculate the order billing amount. In the Order class, add a method to calculate and return the total billing amount for the order. **The billing amount is generally the order amount for an Order plus the computed tax**. However, depending on the job size and order amount, certain discounts may apply.

Based on the job size, customers earn a discount on their orders. Below is a table depicting the discount earned on the order amount based on the job size.

Job Size	**Discount**
'S'	No discount
'M'	1% of the order amount
'L'	2% of the order amount
'X'	3% of the order amount

Additionally, Acme has decided to absorb the tax for high-priced orders. Any order amount greater than $1500 gets this discount.

The formula for an order's total bill is shown below.

Order total = order amount – discount (if applicable) + tax (if applicable)

The method definition should look like the code below.

```
public double computeTotal() {
    //...your work here to compute the total based on the
formula
}
```

1.4 Add method calls on the two order objects in the main method of TestOrders to invoke and test your new methods.

```
System.out.println("The total bill for: " + anvil + " is " +
anvil.computeTotal());
System.out.println("The total bill for: " + balloons + " is " +
balloons.computeTotal());
```

Given the two order objects created in the test, the additional output should look something like the results below.

```
The total bill for: 10 ea. Anvil for Wile E Coyote is 2000.0
The total bill for: 125 ea. Balloon for Bugs Bunny is 1040.0
```

Step 2: Add a leap-year calculator

When taking and processing orders, many Acme employees need to know which years are leap years.

2.1 Add a static method to the MyDate class to list all the leap years between 1752 (the first modern leap year) and 2020. This calculation can be accomplished in a number of ways. Some simple calculators use a basic loop. Other more accurate calculators use a loop and modulus operator, shown in the formula below.

- Every year that is divisible by four is a leap year.
- Of those years, if it can be divided by 100, it is NOT a leap year, unless the year is divisible by 400. Then it is a leap year.

The method should be defined as shown below.

```
public static void leapYears() {
  // ...your code here to compute and println the leap years.
}
```

2.2 Add a method call in the main method of TestMyDate to invoke and test your new method.

```
MyDate.leapYears();
```

Output should look something like that shown below.

```
The year 1752 is a leap year
...
The year 1880 is a leap year
The year 1884 is a leap year
The year 1888 is a leap year
The year 1892 is a leap year
The year 1896 is a leap year
The year 1904 is a leap year
The year 1908 is a leap year
The year 1912 is a leap year
The year 1916 is a leap year
The year 1920 is a leap year
The year 1924 is a leap year
The year 1928 is a leap year
The year 1932 is a leap year
The year 1936 is a leap year
The year 1940 is a leap year
The year 1944 is a leap year
The year 1948 is a leap year
The year 1952 is a leap year
The year 1956 is a leap year
The year 1960 is a leap year
The year 1964 is a leap year
The year 1968 is a leap year
The year 1972 is a leap year
The year 1976 is a leap year
The year 1980 is a leap year
The year 1984 is a leap year
The year 1988 is a leap year
The year 1992 is a leap year
The year 1996 is a leap year
The year 2000 is a leap year
The year 2004 is a leap year
The year 2008 is a leap year
The year 2012 is a leap year
The year 2016 is a leap year
The year 2020 is a leap year
```

2.3 Once the test has passed, you may want to comment out the call to leapYears() to reduce the output in the console during future labs.

Lab Solutions

Order.java

```
package com.acme.domain;

import com.acme.utils.MyDate;

public class Order {
  MyDate orderDate;
  double orderAmount = 0.00;
  String customer;
  String product;
  int quantity;

  public static double taxRate = 0.05;

  public static void setTaxRate(double newRate) {
    taxRate = newRate;
  }

  public static void computeTaxOn(double anAmount) {
    System.out.println("The tax for " + anAmount + " is: " +
anAmount
        * Order.taxRate);
  }

  public Order(MyDate d, double amt, String c, String p, int q)
{
    orderDate = d;
    orderAmount = amt;
    customer = c;
    product = p;
    quantity = q;
  }

  public String toString() {
    return quantity + " ea. " + product + " for " + customer;
  }

  public double computeTax() {
    System.out.println("The tax for this order is: " +
orderAmount
        * Order.taxRate);
    return orderAmount * Order.taxRate;
  }

  public char jobSize() {
    if (quantity <= 25) {
      return 'S';
    } else if (quantity <= 75) {
      return 'M';
    } else if (quantity <= 150) {
      return 'L';
    }
    return 'X';
  }

  public double computeTotal() {
    double total = orderAmount;
    switch (jobSize()){
    case 'M': total = total - (orderAmount * 0.01);
    break;
    case 'L': total = total - (orderAmount * 0.02);
    break;
    case 'X': total = total - (orderAmount * 0.03);
    break;
    }
    if (orderAmount <= 1500){
      total = total + computeTax();
    }
    return total;
  }
}
```

TestOrders.java

```
package com.acme.testing;
import com.acme.domain.Order;
import com.acme.utils.MyDate;

public class TestOrders {

  public static void main(String[] args) {
    MyDate date1 = new MyDate(1,20,2008);
    Order anvil = new Order(date1, 2000.00, "Wile E Coyote",
"Anvil",
       10);

    MyDate date2 = new MyDate(4,10,2008);
    Order balloons = new Order(date2, 1000.00, "Bugs
Bunny","Balloon",
       125);

    System.out.println(anvil);
    System.out.println(balloons);

    System.out.println("The tax Rate is currently: " +
Order.taxRate);
    Order.computeTaxOn(3000.00);
    anvil.computeTax();
    balloons.computeTax();

    Order.setTaxRate(0.06);
    System.out.println("The tax Rate is currently: " +
Order.taxRate);
    Order.computeTaxOn(3000.00);
    anvil.computeTax();
    balloons.computeTax();
    System.out.println("The total bill for: " + anvil + " is " +
       anvil.computeTotal());
    System.out.println("The total bill for: " + balloons + " is
" +
       balloons.computeTotal());
  }
}
```

MyDate.java

```
package com.acme.utils;

public class MyDate {
  // Member/instance variables (a.k.a.
fields/properties/attributes)
  public int day;
  public int month;
  public int year;

  // Constructors:
  // 1. Same name as the class
  // 2. No return type

  // The no args constructor
  public MyDate() {
  }

  // Constructor that takes 3 arguments
  public MyDate(int m, int d, int y) {
    setDate(m, d, y);
  }

  // Methods
  public String toString() {
    return month + "/" + day + "/" + year;
  }

  public void setDate(int m, int d, int y) {
    day = d;
    year = y;
    month = m;
  }

  public static void leapYears() {
    for (int i = 1752; i <= 2020; i = i + 4) {
      if ((i % 100 != 0) || (i % 400 == 0))
        System.out.println("The year " + i + " is a leap year");
    }
  }
}
```

TestMyDate.java

```
package com.acme.testing;
import com.acme.utils.MyDate;

public class TestMyDate{

  public static void main(String[] args){
    MyDate date1 = new MyDate(11,11,1918);

    MyDate date2 = new MyDate();
    date2.day = 11;
    date2.month = 11;
    date2.year = 1918;

    MyDate date3 = new MyDate();
    date3.setDate(4,21,1968);

    String str1 = date1.toString();
    String str2 = date2.toString();
    String str3 = date3.toString();

    System.out.println(str1);
    System.out.println(str2);
    System.out.println(str3);

    MyDate.leapYears();
  }
}
```

Give Intertech a call at **1.800.866.9884** or visit Intertech's website.